UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

Trabajo final del Grado en Ingeniería Informática:

Aplicabilidad de Gazebo en la simulación de vehículos autónomos

Presentado por Carmelo Basconcillos Reyero
en septiembre de 2017
Tutor D. Jesús Enrique Sierra García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Jesús Enrique Sierra García, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos

Expone:

Que el alumno D. Carmelo Basconcillos Reyero, con DNI 71290363T, ha realizado el Trabajo final del Grado en Ingeniería Informática titulado: Aplicabilidad de Gazebo en la simulación de vehículos autónomos.

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a 18 de septiembre de 2017

D. Jesús Enrique Sierra García

Índice de contenido

Índice de ilustraciones

Índice de tablas

Resumen

La robótica ha causado un gran impacto en el mundo actual, tanto es así, que a ninguno nos extraña ver robots en nuestra vida cotidiana, ya sea un aspirador que nos limpia la casa y cuando queda poca batería vuelve a cargar solo, o bien un robot que nos ayude con las labores de cocina… De la misma manera que la robótica aparece en el sector servicios, los robots también están presentes en el sector de la industria, por lo que la empresa ASTI se dedica a crear elementos automatizados como vehículos de guiado automático facilitando así un transporte intralogistico flexible, eficiente y eficaz.

A estos vehículos de guiado automático se les denomina AGV (automatic guided vehicles). Su función es hacer más fácil el trabajo dentro de una empresa o almacén ya que se programan directamente y sus pruebas son realizadas sobre los mismos. Por contrapartida sus inconvenientes son:

- Falta de stock
- Daños en AGVs por una mala programación
- No disponibilidad de un entorno similar al deseado por el cliente

Es aquí donde nace nuestro proyecto, la solución a los problemas descritos serian solucionados con un testeo de estos robots en un simulador.

El objetivo de este proyecto es aprender a desarrollar un entorno que incluya la programación del robot, visualizando así los comportamientos de los AGVs en función a la planificación marcada.

Para la comunicación con el robot, el sistema más extendido es ROS. En los últimos años este ha crecido llegando a incluir una gran comunidad de usuarios en todo el mundo. Históricamente la mayoría de los usuarios estaban en laboratorios de investigación, pero cada vez más vemos como su adopción en el sector comercial. Especialmente en robótica industrial y de servicios.

Palabras Clave

“Robótica, controlador PID, Gazebo, ROS, curso, SDF, modelado, ASTI”

Abstract

Robotics has caused a great impact in the world today, so much so, that we are not surprised to see robots in our daily life, whether a vacuum cleaner that cleans the house and when there is little battery recharges alone, or a Robot to help us with the kitchen work ... In the same way that robotics appears in the service sector, robots are also present in the industry sector, so the company ASTI is dedicated to creating automated elements as vehicles Automatic guidance facilitating flexible, efficient and effective intralogistic transport.

These automatic guiding vehicles are called AGV (automatic guided vehicles). Its function is to make work easier within a company or warehouse since they are programmed directly and their tests are carried out on them. On the other hand, its disadvantages are:

- *Lack of stock*
- *Damage to AGVs due to poor programming*
- *No availability of an environment similar to the one desired by the client*

It is here that our project is born, the solution to the problems described would be solved with a test of these robots in a simulator.

The objective of this project is to learn to develop an environment that includes the programming of the robot, thus visualizing the behaviors of the AGVs in function of the marked planning.

For communication with the robot, the most widespread system is ROS. In recent years this has grown to include a large community of users around the world. Historically the majority of users were in research laboratories, but more and more we see as their adoption in the commercial sector. Especially in industrial robotics and services.

Key words

"Robotic, PID controller, Gazebo, ROS, course, SDF, modelling, ASTI"

I - INTRODUCCIÓN

Este proyecto es un trabajo que se realizará en colaboración con la empresa ASTI, concretamente con el tutor de mi proyecto Don Jesús Enrique Sierra Garcia, profesor de la Universidad de Burgos, y trabaja en el departamento de I+D de ASTI.

El problema planteado consiste en determinar si en dicha empresa sería viable realizar simulaciones en ordenadores sobre los robots que ellos mismos producen. Esto con el fin de determinar las ventajas o inconvenientes que puede producir su desarrollo.

En primer lugar deberiamos abordar el tema de la incrusión del robot dentro del propio simulador, para más tarde comunicarnos con él, y poder manejarlo de manera remota.

Como solución se propone el simulador Gazebo. Su principal ventaja es que se trata de un programa de uso gratuito y su comunidad es bastante grande, por lo que aparecen nuevos modelos de robots.

Realizaremos diferentes pruebas con situaciones adversas para el robots para determinar si el robot se comporta de una manera previsible, probaremos con varios robots para aprender el funcionamiento del programa, así como de la comunicación, para finalmente realizar un estudio sobre este sistema dentro de la empresa.

El objetivo es que de la misma manera que nos comunicamos con el robot que se encuentra dentro de nuestro simulador, en un futuro una vez comprobemos si es viable usar este simulador, podamos usarlo con uno real.

Para la comunicación con el robot usaremos un sistema operativo para robots denominado ROS. Es un sistema operativo de código abierto donde existe una comunicación (publicador/subscriptor) entre nodos, y de esta manera poder ejecutar ordenes.

Más tarde de manejar toda está tecnología deberemos ver la viabilidad que tiene implantar está tecnología en la empresa ASTI, y para concretar mejor los puntos importantes llevaremos a cabo un informe asesorando a dicha empresa sobre el uso de la simulación ROS-Gazebo en su empresa.

II - OBJETIVOS DEL PROYECTO

En este apartado se explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto.

1. Objetivos

Se van a mostrar los puntos que se quieren tratar en este proyecto. Para ello el tutor del proyecto Don Jesús Enrique García de la empresa ASTI, nos plantea un problema a analizar sobre la aplicabilidad de gazebo-ROS en la compañía con el fin de evaluar su viabilidad. En principio los objetivos serian:

- Modelar nuestro robot dentro de la plataforma Gazebo
    - Obtener un primer diseño del robot
    - Identificar sus partes
    - Modelar el diseño a ordenador poniendo los tipos de objeto en función a su utilidad
    - Probar el funcionamiento del robot en la plataforma Gazebo para su correcto funcionamiento

- Conectarnos con el robot creado por medio de ROS
    - Crear un mundo con sus elementos esenciales
    - Lanzar nuestro robot dentro del mundo
    - Lanzar nodos suscriptor y publicador
    - Realizar programación para movimiento del robot
    - Almacenar ordenes en scritpts con movimientos

- Realizar una serie de pruebas documentadas:

  Realizaremos pruebas del funcionamiento y del desarrollo de los robots que servirán para futuros trabajos del mismo área, agilizando así el trabajo de terceros.

- Realizar un informe final:

  En este informe deberemos valorar todo lo que hemos aprendido, evaluando sus posibles ventajas:

    - Analizar las características de la empresa y el impacto que podría tener la inclusión de este sistema en la misma
    - Hacer un balance económico
    - Aconsejar a la empresa sobre su implantación

2. Resumen

En lineas generales las primeras semanas serán llevadas a cabo para documentarnos y aprender como funciona el simulador, realizaremos distintos tutoriales y pruebas, y decidiremos cuál será el entorno de trabajo con el que seguir. Aquí desarrollaremos unos modelos de robots en función a los desarrollados en la empresa ASTI, y realizaremos diferentes pruebas sobre ellos. Una vez elegido ROS deberemos instalar todas sus librerías, aprender a manejar este complejo sistema operativo.

Finalmente es importante elaborar un informe en el cual saquemos una buena y estudiada recomendación para la empresa ASTI, con la que ellos puedan decidir si desean implantar dicha tecnología o no. Para facilitar el trabajo a otras personas que deseen aprender en este mundo ROS dejaremos una serie de pruebas documentadas en nuestra máquina virtual.

III - CONCEPTOS TEÓRICOS

Describiremos los conceptos básicos para poder entender nuestro proyecto.

1. Concepto básicos de Gazebo

Gazebo es el simulador que hemos elegido para desarrollar nuestros modelos, así como todo el comportamiento físico. Como muestra el siguiente gráfico, para diseñar en él debemos tener en cuenta las 3 dimensiones donde z, representa el alto, x representa el ancho e y representará la profundidad. [1]

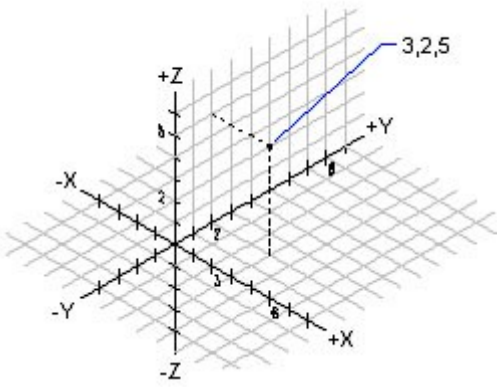

Ilustración 1: Eje coordenadas

Para crear nuestros entornos en este mundo deberemos de tener en cuenta que en el simulador debemos incluir una gravedad para que los objetos no se caigan, y un sol que de luminosidad y sombra a los objetos. Estos objetos (ya sean robots, cuerpos inmóviles o cajas) tienen que tener unas respuestas físicas en nuestro mundo. Es necesario indicar que nuestros robots son cuerpos rígidos y que no son estáticos, y que sufrirán modificaciones cuando choquen con otros, también será importante darle unos valores máximos de velocidad, una masa, una fricción entre otros aspectos.

Más tarde, en los casos de los objetos móviles deberemos añadir una colisión para establecer cuales son los limites del objeto y a partir de que posición el objeto sufrirá un contacto, para ello habrá que marcarle una colisión y la masa del cuerpo. Pero en cambio si queremos crear una pared simplemente le ponemos una colisión y adaptamos los bordes al tamaño de la misma ya que esta no sufrirá cambio físico.

Para diseñar deberemos de tener en cuenta que la computadora debe hacer cálculos para cada colisión, por lo que si llenamos nuestro mundo de colisiones puede que logremos un peor rendimiento, así que se aconseja optimizar y solo poner las necesarias.

2. Concepto básicos de ROS

ROS(Robot Operating System) es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo pero sin serlo puesto que se instala sobre otro(se denomina Meta-Sistema Operativo). ROS se compone de un conjunto de librerías de programación, aplicaciones, drivers y herramientas de visualización, monitorización, simulación y análisis, todas reutilizables para el desarrollo de nuevas aplicaciones para robots tanto simulados como reales. Posee su propio manejador de paquetes mediante comandos desde terminal para la gestión, compilación y ejecución de archivos, así como abstracción de hardware. La siguiente imagen nos ayuda a entender la relación entre ROS y el sistema operativo.

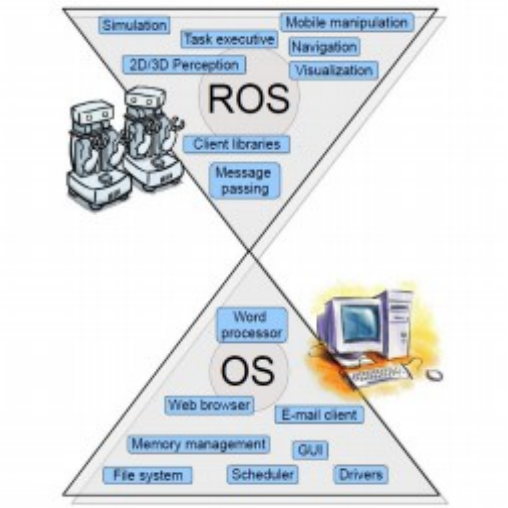

Ilustración 2: Relación ROS y Sistema operativo

Máster: Es un nodo principal que funciona de núcleo del sistema, proporciona registro de nombres y la búsqueda para el resto de los nodos. Hace las veces de plataforma por la que se comunican los diferentes nodos entre si mediante mensajes.

Nodos: Los nodos son procesos independientes que realizan la computación. Los nodos siempre dependen de un Máster, y cada uno desarrolla una función concreta. Por ejemplo, un nodo se encarga de recibir parametros del robot, otro nodo se encarga de mostrarnos información de un sensor del robot, un nodo manda mover un robot, y así sucesivamente. Para escribir un nodo de ROS se utilizan las siguientes librerías cliente:

roscpp: Programación en C++.

rospy: Programación en Phyton.

Mensajes: Los mensajes son una estructura de datos compuestos. Los mensajes son combinaciones de tipos primitivos y mensajes, y sirven para que los nodos se comuniquen entre sí. Para enviar mensajes tienen que ser del tipo standar_msgs.

Tópicos: Los tópicos son canales de comunicación para transmitir datos. Los nodos se suscriben a los tópicos solo si tiene el mismo tipo de mensaje. Cada tópico puede tener varios nodos suscritos. El transporte basado en TCP/IP es conocido como TCPROS y utiliza TCP/IP para la conexión. Este es el tipo de transporte utilizado en ROS. [2]

2.1. Comandos de ROS útiles

La herramienta rosnode es una herramienta de línea de comando para mostrar información sobre los nodos:

rosnode: Muestra información de un nodo reciente.

rosnode info node: Muestra información sobre el nodo.

rosnode kill node: Mata el nodo o envía una señal para matarlo.

rosnode list: Muestra una lista con los nodos activos.

ROS tiene una herramienta para trabajar con tópicos llamada rostopic. Es una herramienta de línea de comandos que proporciona información sobre el tópico o publica datos directamente sobre la red:

rostopic bw /topic: Muestra el ancho de banda utilizado por un tópico

rostopic echo /topic: Muestra el mensaje por la salida estándar.

rostopic info /topic: Muestra información sobre el tópico, el tópico publicado, los que están subscritos y los servicios.

rostopic list: Muestra información sobre los tópicos activos.

rostopic type /topic: Muestra el tipo de tópico, es decir, el tipo de mensaje que publica.

Para crear, modificar y trabajar con paquetes, ROS proporciona algunas herramientas para asistencia:

rospack: Este comando se usa para obtener información o buscar información sobre un paquete

roscreate-pkg: Crear un nuevo paquete utiliza este comando

rosmake: Compilar un paquete

rosdep: Instalar en el sistema las dependencias de un paquete[3]

3. Controlador PID

Un controlador PID (proporcional-integral-derivativo) es un mecanismo de control por re-alimentación. Este calcula la desviación o error entre un valor medido y un valor deseado. El propio controlador intenta minimizar el error a lo largo del tiempo.

En este proyecto se usa para hacer que el robot ajuste su movimiento lo más posible a una trayectoria prefijada, utilizando las variables PID para su regulación.

El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional depende del error actual. El Integral depende de los errores pasados y el Derivativo es una predicción de los errores futuros. La suma de estas tres acciones es usada para ajustar ajustar la velocidad actual, y reducir el fallo.

3.1. Proporcional

La parte proporcional es el control principal, su función es crear una respuesta en la salida proporcional al error, es decir, lograr que el error en estado estacionario se aproxime a cero. La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo, es incluyendo y configurando las acciones integral y derivativa.

La fórmula del proporcional está dada por:

$$P_{\text{sal}} = K_p\, e(t)$$

Ilustración 3: Fórmula del cálculo proporcional

Kp constante de proporcionalidad: se puede ajustar como el valor de la ganancia del controlador o el porcentaje de banda proporcional.

3.2. Integral

El control integral tiene como objetivo disminuir y eliminar el error residual provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por el control proporcional. Solo actuará cuando hay una desviación entre la variable y el punto deseado, integrando esta desviación en el tiempo y sumándola a la acción proporcional.

La fórmula de la integral está dada por:

$$I_{\text{sal}} = K_i \int_0^t e(\tau)\, d\tau$$

Ilustración 4: Fórmula del cálculo integral

Ki constante de integración: indica la velocidad con la que se repite la acción proporcional.

3.3. Derivativo

La acción derivativa solo se produce cuando hay un cambio en el valor absoluto del error; en caso de que el error sea constante, solamente actúan los modos proporcional e integral.

Se utiliza para eliminar las sobreoscilaciones generando una corrección en la señal de control proporcional al error. El objetivo de la acción derivativa es evitar que el error se incremente, para ello mantiene el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce.

La fórmula del derivativo está dada por:

$$D_{\text{sal}} = K_d \frac{de}{dt}$$

Ilustración 5: Fórmula del cálculo derivativo

El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones.

Kd constante de derivación: hace presente la respuesta de la acción proporcional duplicándola, sin esperar a que el error se duplique. El valor indicado por la constante de derivación es el lapso durante el cual se manifestará la acción proporcional correspondiente a 2 veces el error y después desaparecerá.

Tanto la acción Integral como la acción Derivativa, afectan a la ganancia dinámica del proceso. Mientras que la acción integral sirve para reducir el error estacionario, que existiría siempre si la constante Ki fuera nula.

La salida de estos tres términos, el proporcional, el integral, y el derivativo son sumados para calcular la salida del controlador PID[4]. Definiendo y (t) como la salida del controlador, la forma final del algoritmo del PID es:[5]

$$\mathrm{y(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de}{dt}$$

Ilustración 6: Fórmula del cálculo del PID

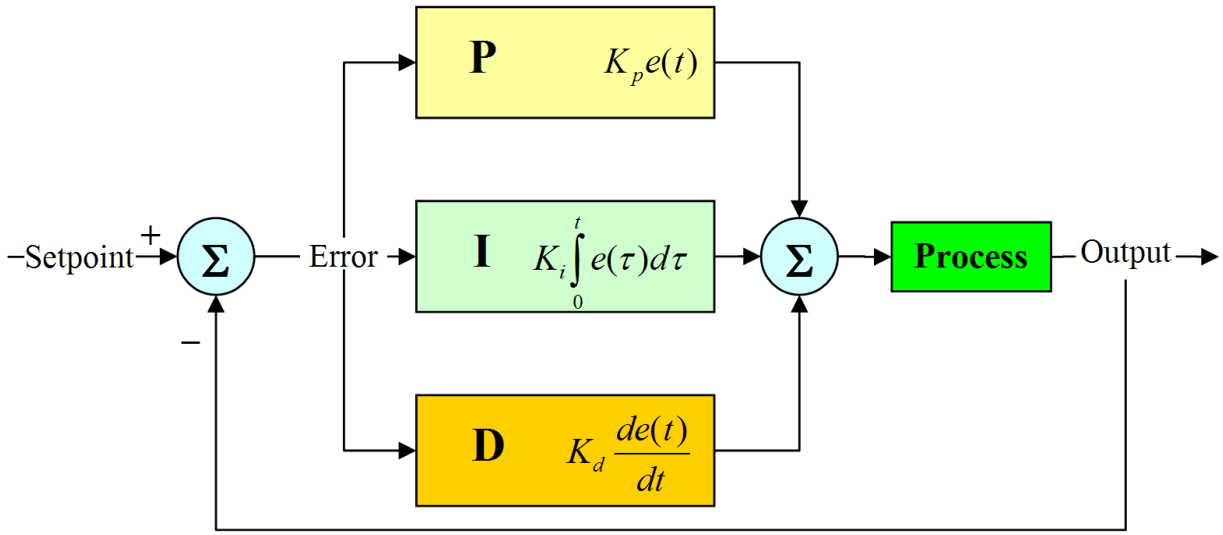

Ilustración 7: Esquema general de un controlador PID

IV - TÉCNICAS Y HERRAMIENTAS

En esta sección vamos a hablar de las distintas técnicas y herramientas que hemos usado y analizado, mostrando las razones que nos han influenciado para tomar la decisión de usarlo. Dada la gran cantidad de herramientas y técnicas para resolver los subproblemas de nuestro proyecto, hemos ido escogiendo varias de las herramientas disponibles y hemos elegido lo que mejor se ajustaba a nosotros. También trataremos los lenguajes usados.

Primero se explicará que técnica o herramienta se ha usado, y luego se definirán otras que se han descartado y los motivos.

1. SDF

Es un formato XML que describe objetos y entornos para simuladores de robots, visualización y control. En sus inicios era desarrollado como parte del simulador Gazebo, pero ahora se distribuye de manera independiente. SDF es un formato estable, robusto y extensible capaz de describir todos los aspectos de robots, objetos estáticos y dinámicos, iluminación, terreno e incluso física.

Además de los atributos cinemáticos y dinámicos, se pueden definir sensores, propiedades superficiales, texturas, fricción de las articulaciones y muchas más propiedades para un robot. Estas características le permiten usar SDF para simulación, visualización, planificación de movimiento y control de robot.

Se ha usado como lenguaje para desarrollar robots propios, o para modificar partes de ya creados en Gazebo. [6]

2. Gestores de versiones

2.1. GitHub

Es un repositorio de versiones donde vamos fijando nuevos issues(tareas a realizar) dentro de un milestone (tarea global a realizar) el código queda organizado en versiones, cada vez que hacemos un commit se actualizaran, y los elementos que modificados aparecen como tal. Se ha usado este repositorio porque se se complementa en una misma aplicación con ZenHub, y entre ambos se ha podido llevar a cabo un seguimiento del trabajo diario,

Ventajas:

- Es de los repositorios mas usados y esta basado en Git.
- Muy útil para el seguimiento de proyectos, cualquiera que vea tu proyecto puede proponerte alternativas o nuevas soluciones.
- Hemos visto GitHub a lo largo de la carrera,por lo que facilita su manejo, es por eso que hemos elegido este.

Desventajas:

- Problemas cuando quieres actualizar más de 100 archivos desde la interfaz a veces da fallo.[7]

Podemos acceder a el a través del siguiente enlace: https://github.com/

2.2. GitKraken

Es un repositorio que funciona con Git su versión gratuita es bastante completa, tiene una interfaz muy vistosa y su manejo es rápido e intuitivo.

Ventajas:

- Tiene un práctico botón con para deshacer operaciones, se pueden revisar errores al momento, lo que hace más fluido el trabajo.
- Es un repositorio muy usado y que esta basado en Git que es casi la referencia en este tipo de proyectos.

Desventajas:

- No se puede incluir mas de veinte personas en los proyectos gratuitos.[8]

Podemos acceder a el a través del siguiente enlace: https://www.gitkraken.com/

3. Gestor de tareas

3.1. ZenHub

Es una herramienta que trabaja con tablones dinámicos, permite la subida de ficheros, y se puede incluir en nuestro repositorio Github. Se ha usado para controlar las tareas de nuestro proyecto, y llevar un seguimiento del mismo en su tablón de tareas.

Ventajas:

- Su principal ventaja es poder integrarlo en github, por lo que podremos trabajar en una sola aplicación.
- Esta ha sido la razón principal por la que he elegido este método.

Desventajas:

- No podremos añadir código pero tampoco es algo irremediable, ya que justo en el repositorio donde se integra la herramienta podemos visualizar dicho código.[9]

Podemos acceder a el a través del siguiente enlace: https://www.zenhub.com/

3.2. Trello

Es un software de administración de proyectos con interfaz web en la cual podemos organizar nuestros proyectos de forma fácil e intuitiva desde cualquier equipo. Nuestra información se guarda en la nube, y podemos accediendo a nuestra cuenta acceder desde cualquier equipo.

Ventajas:

- Como principal punto a favor diré que su curva de aprendizaje es corta, por lo que es fácil aprender a manejarse, además hemos tenido asignaturas en la carrera en las cuales hemos precisado de su uso

Desventajas:

- La principal desventaja por lo cual lo descartaremos es que no viene adjunto a nuestro repositorio, por lo que buscaremos otra solución que implemente esto.[10]

Podemos acceder a el a través del siguiente enlace: https://trello.com/

3.3. Wunderlist

Es una aplicación multiplataforma en el cual podemos gestionar todas las tareas de nuestros proyectos, cada tarea permite añadir un recordatorio, agregar subtareas, notas, o añadir comentarios para un trabajo corporativo.

Ventajas:

- La versión gratuita es muy completa y su manejo muy sencillo e intuitivo.
- Tiene muchas funciones pero a su vez cuentan con una buena usabilidad.

Desventajas:

- Su interfaz es un poco antigua, y necesitaría una actualización.[11]

Podemos acceder a el a través del siguiente enlace: https://www.wunderlist.com/es/

4. Maquina virtual

4.1. VMWare

Es una herramienta de virtualización propiedad de Dell pero cuenta con una versión gratuita con una serie de limitaciones, su principal característica es que sus instrucciones se ejecutan sobre el hardware físico, mientras que otras lo hacen mediante la llamada al sistema operativo anfitrión.

Es una herramienta de virtualización, su principal característica es que sus instrucciones se ejecutan sobre el hardware físico, mientras que otras lo hacen mediante la llamada al sistema operativo anfitrión. Se ha elegido VMWare porque funciona mucho más rápido y sufre menos cuelgues, aunque inicialmente se empezó con VirtualBox por ser la más conocida por lo aprendido en clase. Se ha usado un sistema un sistema Ubuntu dentro de está máquina virtual donde desarrollaremos nuestro proyecto.

Ventajas:

- Es más rápido que VirtualBox comparado en diferentes comparativas de Internet.
- Después de probar con ambos, sufría menos cuelgues al ejecutar programas de gran uso gráfico.

Desventajas:

- La principal desventaja es que se trata de una versión gratuita, y cuenta con una serie de restricciones (como no poder cambiar el numero de procesadores a una maquina creada). [12]

Podemos acceder a el a través del siguiente enlace: http://www.vmware.com/

4.2. VirtualBox

Es una herramienta de virtualización donde podemos crear maquinas virtuales donde instalamos el sistema operativo invitado dentro del sistema operativo de nuestro equipo.

Tiene un manejo fácil y nos permite modificar las características de nuestra maquina virtual después de ser creada.

Ventajas:

- Es propiedad de la empresa Oracle, pero de código abierto, por lo que es gratuita.
- La hemos usado durante la carrera en numerosas asignaturas.

Desventajas:

- Bastante lenta en ejecución.[13]

Podemos acceder a el a través del siguiente enlace: https://www.virtualbox.org/

4.3. Simulador Gráfico

4.4. Gazebo

Es un simulador de múltiples robots, desarrolla un mundo tridimensional en el que poder simular robots, sensores y objetos. Tienen modelos ya diseñados de robots y sensores con los que realizar las simulaciones. Se ha usado Gazebo como simulador para simular nuestros robots y nuestros entornos porque además de ser la única opción gratuita, es la propuesta por el tutor para este proyecto.

Ventajas:

- Las conexiones entre los distintos dispositivos se realiza a través de direcciones
- IP asignándole un puerto a cada uno, de este modo se puede enviar señales a cada robot
- individualmente.
- Es un programa gratuito.

Desventajas:

- No funciona correctamente en Windows.

Podemos acceder a el a través del siguiente enlace: http://gazebosim.org/

4.5. Roboworks

Es un simulador que como Gazebo se basa en la idea de la simulación tridimensional, y en proporcionar una interfaz3D a LabView, Matlab, y Visual Basic. Los modelos se generan gráficamente, y se controla su simulación por medio de archivos de datos, con el teclado, con la interfaz con LabView, Matlab o Visual Basic; o con Robotalk (software que nos permite, remotamente y desde cualquier terminal con TCP/IP, controlar la simulación).

Ventajas:

- Fácil utilización lo que constituye una de sus grandes bazas.
- Dispone de un completo tutorial en español.

Desventajas:

- Solo disponible para Windows 95/98/NT/2000.
- Su precio es de 540€.

Podemos acceder a el a través del siguiente enlace: http://www.newtonium.com/

4.6. Webots

Es un simulador que permite la simulación de numerosos tipos de robots, incluidos robots definidos por el usuario. Incluye completas librerías con todo tipo de actuadores y sensores. Permite la programación del robot mediante C, C++ y Java.

Su precio varía desde los 320 euros de la versión de educación, hasta los 3500 euros de la versión profesional.

Ventajas

- Utiliza las librerías Open Dynamics Engine (ODE, que son de código abierto)
- para las simulaciones físicas, por lo que podremos modificarlas a nuestro gusto.
- Permite guardar las simulaciones en formato de vídeo AVI o MPEG.
- Incluye ejemplos
- Disponible para los sistemas operativos Linux, Windows, y Mac OS X.

Desventajas

- Su elevado precio desde los 320 euros de la versión de educación(versión reducida), hasta los 3450 euros de la versión profesional.[14]

Podemos acceder a el a través del siguiente enlace: https://www.cyberbotics.com/

Comparativa de los 3:

	Coste licencia	Robots integrados	Entorno editable	Diseñar propios robots	Lenguajes
Gazebo	Gratuito	Si	Si	Si	C, C++
Roboworks	540€	Si	No	Si	LavView, Matlab, VB
Webots	3500€(versión pro)	Si	Si	Si	C, C++ C#, VB#, J#

Tabla 1: Comparativa simuladores

5. Modelado

El modelado consiste en la creación de diagramas que nos explica las relaciones que van a tener los objetos entre ellos dentro de nuestra aplicación.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo.

5.1. Astah

Es una herramienta de modelado UML para la creación de diagramas orientados a objetos. Se ha usado está herramienta para desarrollar los diferentes diagramas referentes a los anexos, su fácil manejo, y navegabilidad ha hecho que sea la mejor opción.

Ventajas

- Programa muy completo y sencillo de usar
- Fácil navegabilidad entre los diferentes diagramas
- Lo hemos usado en la carrera por lo que su uso es más sencillo
- Fáciles videotutoriales para aprender su manejo dentro de su pagina
- Hemos usado este programa por todas estas ventajas

Desventajas

- Su mayor desventaja es que su licencia cuesta 1190$ para un año, o 2490$ de manera continua, para hasta 10 trabajadores, aunque cuenta con la versión estudiante que usaremos.[15]

Podemos encontrar la herramienta a través del siguiente enlace http://astah.net/.

5.2. Dia

Es una aplicación para la creación de diagramas. El formato para leer y almacenar gráficos es XML. Puede producir salida en diferentes formatos.

Ventajas

- Programa gratuito
- Tiene un fácil aprendizaje, y sencilla interfaz
- Tiene cabida en windows, linus y OS X.

Desventajas

- Programa más simple, y menos completo que el anterior
- No podemos realizar todos los tipos de diagramas

Podemos acceder a la pagina atraves del siguiente enlace http://dia-installer.de/index.html.es

6. Sistema Operativo para Robot

6.1. ROS

ROS (Robot Operating System) es una plataforma de desarrollo open source para sistemas robóticos. Proporciona toda una serie de servicios y librerías que simplifican considerablemente la creación de aplicaciones complejas para robots. Se ha usado ROS como sistema de comunicación con nuestros robots porque además de ser una opción que funciona perfectamente con el simulador Gazebo.

Ventajas

- Permite reutilizar código
- Sus amplias librerías agilizan el trabajo
- Paso de mensaje entre procesos
- Elegimos esta opción puesto que es la más extendida, la más conocida, y es de código abierto

Desventajas

- Curva de aprendizaje es muy lenta
- Solo disponible para Linux

Podemos acceder a la pagina a traves del siguiente enlace http://www.ros.org/

6.2. Microsoft Robotics Developer Studio 4

Es un entorno de programación basada en .NET libremente disponible para la creación de aplicaciones de robótica. Puede soportar una amplia gama de plataformas robóticas, ya sea corriendo directamente sobre la plataforma o controlar el robot desde un PC con Windows a través de un canal de comunicación, tales como Wi-Fi o Bluetooth.[16]

Ventajas

- Contiene una extensa documentación y una amplia serie de muestras y tutoriales
- Gran numero de robots compatibles

Desventajas

- Solo disponible para Windows
- Curva de aprendizaje es muy lenta

Podemos encontrar la herramienta a traves del siguiente enlace

7. Lenguaje de ROS

Los lenguajes soportados por ROS para crear librerias para los robots son únicamente 2, analizamos ambos.

7.1. C++

C++ es un lenguaje de programación diseñado para extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Usaremos este lenguaje porque es el que interpreta el sistema operativo ROS.[17] En cuanto a la programación en C++ nos basamos en ejemplos facilitados por Raúl Trapiela, experto en ROS que nos ha ayudado en la programación con C++.[18]

7.2. Python

Python es un lenguaje de programación muy intuitivo. Se trata de un lenguaje que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

8. Visualizador de fuerzas

8.1. Rviz

Es un visualizador 3D que nos muestra representaciones en vivo de los valores de los sensores lanzados por nodos ROS, como datos de la cámara, mediciones de distancia infrarroja, etc...Se ha usado esta aplicación para ver fuerzas y movimientos que se producen sobre nuestros modelos.

9. Grabador de vídeo

9.1. SMRecorder

Es una aplicación que se usa para capturar vídeos con audio del escritorio. Se ha usado este grabador de vídeo para poder grabarnos haciendo diferentes pruebas con los robots, su fácil uso, y su sencillez ha hecho que sea la mejor elección

Ventajas:

- Nos permite exportar a cualquier tipo de archivo
- Hemos seleccionado este porque el otro no funcionaba correctamente
- Sencillo, y da buen resultado

Desventajas:

- Graba desde el momento que abres el programa, y luego siempre necesitarías editar los vídeos[19]

9.2. Presentation Tube

Programa que permite grabar la pantalla de nuestro ordenador a tiempo real mientras podemos interaccionar con el ratón diferentes opciones. Grabará también si lo deseamos el audio que introduzcamos por un micrófono.

Ventajas:

- Muy sencillo de usar
- Es gratuito

Desventajas:

- Solo exporta la versión gratuita en .asf y nos da error al reproducir

V - ASPECTOS RELEVANTES DEL DESARROLLO DEL PROYECTO

En este capítulo vamos a detallar los puntos claves del proyecto y como se han llevado a cabo.

1. Primeros pasos

Los primeros días con este proyecto fueron inciertos, sabíamos lo que queríamos pero no sabíamos como desarrollarlo. Primeramente vimos que el simulador que venia propuesto (Gazebo) solo funcionaba con Linux, por lo que instalamos una versión de Linux en nuestro gestor de máquinas virtuales(VirtualBox), aunque después de ver que muchas veces nos daba fallos lo sustituimos(VMware) por otro que nos dio mejor rendimiento. Ya con nuestro sistema operativo instalado instalamos el simulador Gazebo y empezamos a realizar diferentes tutoriales sobre su funcionamiento.

Insertamos los primeros robots turtlebot, y youbot de la librería Gazebo, y se les dio ordenes de movimiento, se insertaban objetos y se veia su comportamiento antes los impactos.

2. Manejo de SDF

Pronto vimos que podríamos diseñar nuestro propio robot dentro de este simulador, observamos que estos robots ya creados en sus librerias funcionaban con URDF los más antiguos, y SDF los más nuevos y empezamos a desarrollar nuestras primeras formas.

Íbamos aprendiendo a manejar SDF, un formato XML que funciona definiendo diferentes formas y dándoles características propias. Sus tres principales etiquetas son:

- Model: Incluye a todo el modelo, se le da un nombe, una posición en el plano, y si se quiereque sea estático o móvil ese modelo. El modelo siempre debe llevar al menos dos links y un joint.
- Link: Es un objeto visto(una caja, una rueda, un brazo...) tendrá una parte visual y otra de colisión, en ambas deberán de tener su situación en el mapa, su extensión y su masa obligatoriamente, y parámetros de velocidad, fricción, odometria, de manera opcional.
- Joint:Es la unión entre dos links, sieempre deberá indicar que dos links está uniendo, además de suposición en el plano, en caso de ser ruedas, o uniones que no sean fijas(fixed) se puede usar parametros como velocidad máxima a la unión, o ángulo de giro.

Más tarde observamos que podríamos incluir diferentes sensores (ya creados) que se pueden integrar a nuestro robot,y decidimos probarlos y comprobar su funcionamiento. De estos es desde donde puedes sacar los datos para más tarde mediante ROS comunicarte con el, y poder ordenar movimientos en función de los parámetros recibidos de los sensores.

- Include: Sirve para añadir un modelo dentro de tu modelo, por ejemplo incluir un sensor Hokuyo dentro de un modelo creado, donde daremos su posición dentro del modelo, o cualquier otro tipo de complemento(un brazo, una cámara etc).
- Plugin: El plugin sirve para dotar al modelo de una programación anteriormente compilada, declaras la librería que quieres y el robot interpretará las ordenes en función de su programación.

En este punto cabe destacar que manejar bien SDF, y insertar todos sus atributos aunque inicialmente no sean necesarios(ya sean masa, su forma, inercia,velocidad máxima) es importante, porque más adelante vas a poder necesitar de estos para leer algún dato.

3. Herramienta para manejo de de robots empleado

Como ya hemos visto en el punto anterior, hay varias alternativas sobre qué herramienta usar para la comunicación con el robot. En este caso según lo propuesto por el Tutor, empezamos a aprender en el entorno ROS ya que tiene muchas ventajas como ser el más usado en la actualidad, estar en un crecimiento constante y ser completamente gratuito. Hemos realizado diferentes tutoriales y cursos sobre ROS, y hemos aprendido como funciona y como manejar nodos, comunicarnos entre ellos, y diferentes sentencias que nos dan información sobre el estado de nuestros nodos. ROS trabaja con publicadores y suscriptores:

- Para lanzar un nodo publicador debemos de lanzar siempre todas las dependencias de ROS en ese terminal concreto, para ello se usa un setup.bash que se crea al generar el paquete. Y después llamar al comando:

  rostopic echo -c “nombre del topico”

  De esta manera, el robot te comunicará aquello que le hayas programado a decir, por ejemplo a que distancia esta de un objeto, que elementos tipo joint tiene, cuantos objetos reconoce en su radio etc...

- Para lanzar un nodo suscriptor debemos de lanzar siempre todas las dependencias de ROS en ese terminal concreto, para ello se usa un setup.bash que se crea al generar el paquete. Y después llamar al comando:

  rostopic pub “nombre del topico” std_smgs/String “orden a ejecutar”

  Así el robot según la programación que haya sido instalada interpretará la orden y la eje cutará, estas ordenes pueden ser de acercarse o alejarse de un objeto, moverse, mover alguna de sus partes.

4. Lenguaje para el desarrollo del proyecto

Desde un primer momento se decidió el empleo de C++ como lenguaje central en la implementación del proyecto dado que permite trabajar con ROS (la otra opción disponible para ROS es python). Y nos lanzamos a programar con la ayuda de diferentes tutoriales que encontrábamos por Internet, hasta que dimos con Raul Trapiela, un experto en ROS que nos ayudo con la realización de unos tutoriales con los que podes entender toda la comunicación ROS, y facilitarnos un código para conseguir mover los robots.[20]

5. Controlador PID

Controlador que viene incluido por las librerías de Gazebo, debemos usarlo para controlar la fuerza que ejercemos sobre el robot, el mismo hace mediciones y va retroalimentandose haciendo cálculos sobre su movimiento actual y el futuro para dar un movimiento correcto al robot. Este controlador es esencial para conseguir mover los robots, sin el se lanzaría una fuerza al robot y saldría volando por los aires.

De tal manera que cada vez que se quiera aplicar fuerza a algún elemento de los robots se llamará este controlador, dando unos valores a sus tres variables(Proporcional, integral y derivativo), el valor superior e inferior de la integral, y el valor máximo y mínimo de la velocidad que se quiere proporcionar.

Así es, que cada vez que queramos mover un objeto desde ROS se deberá solicitar a este método que controle su velocidad.

6. Pruebas experimentales

Para ver que el robot se comporta de manera correcta hemos realizado diferentes pruebas:

- Superar un cuerpo
- Impactar con cuerpo pesado
- Un mundo sin suelo
- Construcción y muestra de un mundo complejo
- Distintas pruebas con posibles fallos de programación

Más tarde también se han realizado pruebas para conseguir comunicarnos con el robot, de tal manera que en el publicador busques que el robot te diga sus partes o te escriba qué elementos tienen mas de una longitud, en definitiva hacer pruebas para ver como es cada robot. Y como se comportan en cada situación. Para ello se realizarón las siguientes pruebas:

- Insertar Agv y probar cambio dirección
- Mover detener y retroceder y comportamiento ante los objetos
- Mover un objeto después de moverse el robot con un brazo
- Mover dos robots en el mismo entorno

7. Informe

Una vez finalizada la etapa de programación e investigación sobre estas tecnologías nos centraremos en realizar un informe detallado sobre la viabilidad de estas técnicas en una empresa de Robótica como puede ser ASTI, de tal manera que este proyecto pueda servir de material didáctico para ayudar a acortar los plazos de aprendizaje de esta tecnología. Para ello se seleccionaron los puntos claves para el informe:

- Objetivos: Se han marcado las principales ventajas de trabajar con el simulador, y el objetivo es implantar esta tecnología para que se hagan realidad estas ventajas.
- Metodología: Se ha descrito la forma en la cual se ha trabajado en este proyecto
- Contenidos: Se ha realizado una lista con los principales contenidos a tener en cuenta.
- Temporalidad: Se ha realizado una aproximación del tiempo que podría tardar una empresa en implantar esta tecnología.
- Metodología de implantación: Se redacta un plan a seguir para la implantación de este sistema.

Más adelante se desarrollaron los temas de estudio de mercado, donde comparamos con otras empresas del sector, y recursos humanos donde se describía brevemente el grupo de trabajo y la preparación necesaria del grupo de trabajo.

A continuación se desarrollará un detallado plan financiero, donde se tendrán en cuenta:

- Costes de licencias y hardware
- Costes de trabajadores
- Previsión de gastos
- Previsión de ingresos y amortización

Finalmente se ha desarrollado una conclusión con la que se aconseja la implantación de esta tecnología en una empresa de AGV.

VI - TRABAJOS RELACIONADOS

1. Análisis de un brazo robótico con Gazebo y ROS para tareas de inspección remota en el CERN

Este trabajo de Fin de Máster en Automática y Robótica de José Luis Samper Escudero fue hecho para la organización europea para la investigación Nuclear. Trata de mejorar las tareas de inspección y mantenimiento mediante la integración de un manipulador robótico de 6 grados de libertad que desarrollará en Gazebo_ROS, y realizará tareas de mantenimiento que comprenden desde simples inspecciones visuales a complejas labores como puede ser desatornillado o extracción de componentes dañados.

Este proyecto, pese a que la finalidad no es la misma que el mio, se desarrolla de una forma parecida, y los pasos a realizar son similares, usa una metodología similar a la nuestra.

Enlace al trabajo: http://oa.upm.es/39677/1/TFM_JOSE_LUIS_SAMPER_ESCUDERO.pdf

2. Operación de remachado mediante robot manipulador industrial basado en ROS

El trabajo de fin de grado en Ingeniería de las Tecnologías Industriales de Víctor Hugo Gómez Tejada de la Universidad de Sevilla se sitúa en un entorno de simulación donde debemos realizar una tarea de remachado a una pieza de trabajo mediante un robot manipulador industrial. Programa un robot mediante el framework robótico ROS para realizar la tarea de remachado, evaluando su rapidez y eficiencia.

Aunque desde el punto de vista de programación no está muy evolucionado, está muy bien definida toda la parte teórica de los ejes de coordenadas. Tiene un gran trabajo de documentación y explicaciones.

Enlace al trabajo: http://bibing.us.es/proyectos/abreproy/90360/fichero/TFG+V%C3%ADctor+Hugo+G%C3%B3mez+Tejada.pdf+

3. Extendiendo las capacidades del robot RESCUER en ROS

Este trabajo de Julio Jesús León Pérez del Máster universitario en Sistemas Inteligentes de la Universidad Jaume I trata de como un robot físico con el que cuentan en dicha universidad lo usan en distintas plataformas, para poder comprobar su comportamiento tanto real como virtual. En la parte virtual al probar con MoveIt lo hacen con éxito, puesto que la programación hecha da la respuesta esperada, pero en Gazebo les queda bastante trabajo por realizar puesto que el robot se mueve de manera errática.

Es un trabajo similar al nuestro en la parte de Gazebo, donde por medio de ROS implementan un robot creado y le dan movimiento.

Enlace al trabajo: http://repositori.uji.es/xmlui/bitstream/handle/10234/147985/TFM_Leo%CC%81n%20Pe%CC%81rez_Julio%20Jesu%CC%81s.pdf?sequence=1

VII - CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

En este apartado, vamos a mencionar algunas conclusiones a las que hemos llegado tras realizar el proyecto. Se verán las partes más importantes que se han tomado para que futuros desarrolladores puedan continuar el proyecto.

1. Conclusiones

Considero que el proyecto llevado a cabo es muy ambicioso ya que se han abordado múltiples herramientas las cuales desconocíamos y nos han servido para ampliar conocimientos.

En primer lugar nos pusimos a documentarnos sobre la herramienta Gazebo, comenzamos a realizar tutoriales y nos fuimos dando cuenta de que cuanto más aprendíamos se abrían más campos en los que abordar.

Llegados a este punto, vimos que la mejor opción para seguir desarrollando el proyecto era aprender ROS, un sistema operativo para robots que se perfila como una de las mejores opciones en cuanto a la programación robótica.

Se abría de nuevo un nuevo mundo del que no sabíamos nada, empezábamos nuevamente a realizar tutoriales, y aprender de manuales, pero con la gran diferencia, que en el mundo ROS estaba mucho peor documentado.

Para plasmar el proyecto decidimos que la mejor forma era hacer un informe para una empresa de logística interna como ASTI y realizar un curso para acelerar el aprendizaje de terceros.

En general, podemos decir que implantar Gazebo-ROS en las empresas de logística es claramente beneficioso puesto que podemos simular un mundo similar a la realidad y evitar cometer fallos que en la realidad podrían ser fatales, de esta forma que una buena programación de un robot simulado puede servirnos de la misma manera para uno real, por lo que todo serían ventajas.

El sistema ROS se encuentra en constante desarrollo y evolución, por lo que el adecuado estudio del mismo no acaba con este proyecto, sino que representa una tarea permanente.

Los conocimientos adquiridos durante la realización de la carrera universitaria, me han ayudado de gran manera a llevar a cabo el proyecto. Las principales asignaturas sobre las que me he apoyado son:

- Sistemas Inteligentes: Sin duda la asignatura con la que guarda más relación dado que esta asignatura, nos introdujo en los entornos inteligentes, nos enseño a resolver problemas de IA con robots.
- Sistemas Operativos: La navegación por Ubuntu mediante líneas de comandos, así como el manejo de diferentes terminales son conocimientos vistos en la asignatura.
- Gestión de proyectos: Otra de las asignaturas fundamentales en la planificación y gestión del proyecto, las metodologías ágiles han sido necesarias.

2. Lineas de trabajo Futuras

Como futuros proyectos propuestos se podría centrar en para generar el código necesario para mover un robot real utilizando ROS, al que se le pueden ir añadiendo mas funciones con el paso del tiempo. Por supuesto este robot deberá compartir características con los robots diseñados en el simulador, probar que todo el código es completamente funcional, y luego cargarlo en nuestro robot real.

2.1. Robot en un entorno real

Utilizar ROS en un robot ya creado de la galería que dispone Gazebo(o uno de los desarrollados por nosotros), y generar un código válido para poder mover un robot real, más tarde se podría entrar a manejar sus sensores también desde ROS, y así hasta tener el robot real completamente manejado por el sistema para robots ROS.

VIII - REFERENCIAS

Bibliografía

[1] Alexander Subirós Martínez , Aprende autocad 3D, 2006, http://www.mailxmail.com/curso-aprende-autocad-3d/introducir-coordenadas-z
2: Ricardo Ragel de la Torre, Entorno de simulaciónROS/Gazebo, 5 de febrero del 2017, 11:53 UTC
[3] Erle Robotics S.L, Introducción a ROS, 2017, http://erlerobotics.com/blog/ros-introduction-es/
[4] colaboradores de la wikipedia, Controlador PID, 10 de junio del 2017, 09:47 UTC
[5] Iñigo Gútiez, Ejemplo de control de temperatura, 1 de julio del 2017, 10:04 UTC, https://programacionsiemens.com/pid-en-step7/
[6] , API SDF, 14 de marzo del 2017, 10:03 UTC
[7] colaboradores de Wikipedia, GitHub, 30 de junio del 2017, 11:44 UTC, https://es.wikipedia.org/w/index.php?title=GitHub&oldid=98817555
[8] Enric Florit, GitKraken, un nuevo cliente de git multiplataforma y gratuito, 30 de abril del 2017, 11:48 UTC, http://www.enricflorit.com/gitkraken-un-nuevo-cliente-de-git-multiplataforma-y-gratuito/
[9] Juan Diego Polo, zenhub, plataforma de gestión de proyectos integrada en Github, ahora gratis para estudiantes, , https://wwwhatsnew.com/2015/09/24/zenhub-plataforma-de-gestion-de-proyectos-integrada-en-github-ahora-gratis-para-estudiantes/
[10] Fernando Siles, ZenHub.io, vitaminas para tu GitHub, 30 de mayo del 2017, 10:53 UTC, https://www.genbetadev.com/herramientas/zunhub-io-vitaminas-para-tu-github
[11] Miriam Schuager, Wunderlist estrena diseño y nuevas características en su versión web, , https://wwwhatsnew.com/2015/08/14/wunderlist-estrena-nuevo-diseno-en-su-version-web/
[12] colaboradores de Wikipedia, VMWare, 4 de Febrero del 2017, 12:08 UTC
[13] colaboradores de Wikipedia, VirtualBox, 3 de febrero del 2017, 12:06 UTC
[14] Elena Rodrigo López Javier Puertas Torres Diego López García Ramón García Olivares-Francisco Javier Fernández Oscar Yago Corral Guillermo Asín PrietoEsther Samper Domeque, Simuladores en robótica, 3 de mayo del 2017, 16:08 UTC
[15] colaboradores de Wikipedia, Astah, 30 de junio del 2017, 16:17 UTC
[16] Microsoft, Microsoft Robotics Developer Studio, 17 de marzo del 2017, 12:17 UTC, https://www.microsoft.com/en-us/download/details.aspx?id=29081
[17] colaboradores de Wikipedia, C++, 22 de abril del 2017, 12:23 UTC
[18] Raúl Trapiela, C++, 2017, https://github.com/carmelobasconcillos/miRobot
[19] José Aguirre, SMRecorder: Vídeos con audio del escritorio y webcam, 18-02-2011, http://tecnologia21.com/30598/smrecorder-videos-audio-escritorio-webcam
[20] Raúl Trapiela, Robótica, 2017, https://www.youtube.com/playlist?list=PL2bUNdUcvv-m0CtampWejh0ekjAP8QB9bc

Impreso en Burgos el domingo, 17 de septiembre de 2017